IN THE CLAIMS

Please cancel claims 50, 111-112, and 119-125. Please amend claims 43, 52, 54, 56,109-110,113-114,116,118, and 126-128 as indicated.

1-42 (Cancelled)

43. (Currently Amended)A system for executing stack and register based instructions, comprising:

~~a central processing unit (CPU) core~~ execute logic for executing the register-based instructions ;

a register file associated with the ~~CPU core~~ execute logic ; and

a hardware accelerator to process the stack-based instructions in cooperation with the ~~CPU core~~ execute logic, wherein the hardware accelerator marks variables associated with the stack-based instructions in the register file as modified when the variables are updated as a result of the processing of the stack-based instructions to enable selective writing of the variables marked as modified to a memory.

44 -51 (Cancelled)

52. (Currently Amended) A ~~system~~ CPU for executing stack and register-based instructions, comprising:

~~a central processing unit (CPU) core~~ execute logic for executing the register-based instructions;

a register file associated with the ~~CPU core~~ execute logic; and

a hardware accelerator to process the stack-based instructions in cooperation with the ~~CPU core~~ execute logic, wherein the hardware accelerator generates a new virtual machine program counter (PC) due to a "JSR" or "JSR_W" bytecode by sign extending ~~the~~ an immediate branch offset following the "JSR" or "JSR_W" bytecode and adding it to ~~the~~ a virtual machine program counter (PC) of ~~the~~ a current bytecode

instruction, computes ~~the~~ a return virtual machine program counter and pushes the return virtual machine program counter (PC) onto an operand stack.

53. (Cancelled)

54. (Currently Amended) A ~~system~~ central processing unit (CPU) for executing stack and register-based instructions, comprising:

~~a central processing unit (CPU) core~~ execute logic for executing the register-based instructions;

a register file associated with the ~~CPU core~~ execute logic; and

a hardware accelerator to process the stack-based instructions in cooperation with the ~~CPU core~~ execute logic ; wherein the hardware accelerator performs sign extension for ~~the~~ virtual machine SiPush and BiPush bytecodes and appends the sign extended data to ~~the~~ an immediate field of a register-based instruction being composed based on the stack-based instructions.

55. (Cancelled)

56. (Currently Amended) A ~~system~~ central processing unit (CPU) for executing stack and register-based instructions, comprising:

~~a central processing unit (CPU) core~~ execute logic for executing register-based instructions;

a register file associated with the ~~CPU core~~ execute logic ; and

a hardware accelerator to process the stack-based instructions in cooperation with the ~~CPU core~~ execute logic , wherein the hardware accelerator performs sign extension for ~~the~~ virtual machine SiPush and BiPush bytecodes and makes the sign extended data available to be read by the ~~CPU core~~ execute logic.

57-108. (Cancelled)

109. (Currently Amended) The system of claim 43, wherein the hardware accelerator and the ~~CPU core~~ execute logic are within a CPU.

110. (Currently Amended) The system of claim 43, wherein the hardware accelerator processes the stack-based instructions in cooperation with the ~~CPU core~~ execute logic by converting the stack-based instructions into register-based instructions for execution in the ~~CPU core~~ execute logic .

111. (Cancelled)

112. (Cancelled)

113. (Currently Amended) The ~~system~~ CPU of claim 52, wherein the hardware accelerator and the ~~CPU core~~ execute logic are within a CPU.

114. (Currently Amended) The ~~system~~ CPU of claim 52, wherein the hardware accelerator processes the stack-based instructions in cooperation with the ~~CPU core~~ execute logic by converting the stack-based instructions into register-based instructions for execution in the ~~CPU core~~ execute logic .

115. (Cancelled)

116. (Currently Amended) The ~~system~~ CPU of claim 54, wherein the hardware accelerator processes the stack-based instructions in cooperation with the ~~CPU core~~ execute logic by converting the stack-based instructions into register-based instructions for execution in the ~~CPU core~~ execute logic.

117. (Cancelled)

118. (Currently Amended) The ~~system~~ CPU of claim 56, wherein the hardware accelerator processes the stack-based instructions in cooperation with the ~~CPU core~~ execute logic by converting the stack-based instructions into register-based instructions for execution in the ~~CPU core~~ execute logic.

119-125 (Cancelled)

126. (Currently Amended) A ~~system~~ central processing unit (CPU) for executing stack and register-based instructions, comprising:

~~a central processing unit CPU core~~ execute logic for executing the register-based instructions ;

a register file associated with the ~~CPU core~~ execute logic ; and

a hardware accelerator to process the stack-based instructions in cooperation with the ~~CPU core~~ execute logic , wherein the hardware accelerator:

maintains an operand stack for the stack-based instructions in the register file such that the operand stack in the register file defines a ring buffer in conjunction with an overflow/underflow mechanism for moving operands in the operand stack between the register file and memory, and loads variables required for processing the stack-based instructions into the register file[[,]] ;

generates a new virtual machine program counter due to a "GOTO" or "GOTO_W" bytecode by sign extending ~~the~~ an immediate branch offset following the "GOTO" or "GOTO_W" bytecode and adds it to ~~the~~ a virtual machine program counter of ~~the~~ a current bytecode instruction[[,]] ;

generates a new virtual machine program counter due to a Jump sub routine(JSR) or "JSR_W" bytecode by sign extending ~~the~~ an immediate branch offset following the "JSR" or "JSR_W" bytecode and adding it to ~~the~~ a virtual machine program counter (PC) of ~~the~~ a current bytecode instruction, computes ~~the~~ a return virtual machine program counter and pushes the return virtual machine program counter onto the operand stack [[,]] ;

performs a sign extension for ~~the~~ virtual machine SiPush and BiPush bytecodes and appends the sign extended data to ~~the~~ an immediate field of a register-based instruction being composed based on the stack-based instructions[[,]] ;

performs sign extension for ~~the~~ virtual machine SiPush and BiPush bytecodes and makes the sign extended data available to be read by the ~~CPU core~~ execute logic [[,]] ; and

produces exceptions in respect of selected stack-based instructions ~~marks variables associated with the stack-based instructions in the register file as modified~~

~~when the variables are updated as a result of the processing of the stack-based instructions to enable selective writing of the variables marked as modified to a memory~~.

127. (Currently Amended) The ~~system~~ CPU of claim 126, wherein the exceptions are processed using register-based instructions ~~hardware accelerator and the CPU core are within a CPU~~.

128. (Currently Amended) The ~~system~~ CPU of claim 126, wherein the hardware accelerator processes the stack-based instructions in cooperation with the ~~CPU core~~ execute logic by converting the stack-based instructions into register-based instructions for execution in the ~~CPU core~~ execute logic

129. (New) A ~~system~~ central processing unit (CPU), comprising:

execute logic to receive and process input corresponding to register-based instructions;

a hardware accelerator to process stack-based instructions to produce an output that can be processed by the execute logic;

an operand stack for the stack-based instructions, the operand stack being maintained in a register file as a ring buffer;

an overflow/underflow mechanism for moving operands in the operand stack between a register file and a memory, said register file also storing data associated with the register-based instructions;

a bytecode buffer that receives stack-based instructions from the memory;

an instruction decode unit coupled to the bytecode buffer to decode instructions received from the bytecode buffer; and

a bytecode buffer control element that receives an indication of the number of bytecodes used by the instruction decode unit to control loading of the stack-based instructions into the bytecode buffer from the memory.

130. (New) The CPU of claim 129, wherein the instruction decode unit produces an indication for a variable stored in the register file

131. (New) The CPU of claim 129, further comprising a microcode unit coupled to the instruction decode unit to receive output therefrom.

132. (New) The CPU of claim 129, wherein the hardware accelerator produces an exception in respect of selected stack-based instructions.

133. (New) The CPU of claim 129, wherein the instruction decode unit decodes multiple instructions received from the bytecode buffer in parallel.

134. (New) The CPU of claim 131, wherein the instruction decode unit generates a start address for the microcode unit.

135. (New) The CPU of claim 134, further comprising a mechanism to select the start address or an incremented start address for the microcode.

136. (New) The CPU of claim 129, further comprising a common instruction cache for the stack-based instructions and the register-based instructions.

137. (New) The CPU of claim 131, wherein the instruction decode unit comprises multiple decoders.

138. (New) The CPU of claim 134, wherein the microcode unit produces a stack update indication to cause the instruction decode unit to generate a new start address for the microcode unit.

139. (New) A central processing unit (CPU), comprising:

execute logic to receive and process input corresponding to register-based instructions;

a hardware accelerator to process stack-based instructions to produce an output that can be processed by the execute logic;

an operand stack for the stack-based instructions, the operand stack being maintained in a register file as a ring buffer;

an overflow/underflow mechanism for moving operands in the operand stack between a register file and a memory, said register file also storing data associated with the register-based instructions;

a bytecode buffer that receives stack-based instructions from the memory; and

an instruction decode unit coupled to the bytecode buffer that decodes multiple instructions received from the bytecode buffer in parallel.

140. (New) The CPU of claim 139, wherein the instruction decode unit produces an indication for a variable stored in the register file.

141. (New) The CPU of claim 139, wherein the instruction decode unit comprises multiple decoders.

142. (New) The CPU of claim 139, wherein the hardware accelerator produces an exception in respect of selected stack-based instructions.

143. (New) The CPU of claim 139, further comprising a microcode unit coupled to the instruction decode unit to receive output therefrom.

144. (New) The CPU of claim 139, wherein the instruction decode unit generates a start address for the microcode unit.

145. (New) The CPU of claim 144, further comprising a mechanism to select the start address or an incremented address for the microcode unit.

146. (New) The CPU of claim 145, wherein the microcode unit produces a stack update indication to cause the instruction decode unit to generate a new start address for the microcode unit.

147. (New) A central processing unit (CPU) comprising:

execute logic to receive and process input corresponding to register-based instructions;

a hardware accelerator to process stack-based instructions to produce an output that can be processed by the execute logic;

an operand stack for the stack-based instructions, the operand stack being maintained in a register file as a ring buffer;

an overflow/underflow mechanism for moving operands in the operand stack between a register file and a memory, said register file also storing data associated with the register-based instructions;

a bytecode buffer that receives stack-based instructions from the memory; and

an instruction decode unit coupled to the bytecode buffer to decode instructions received from the bytecode buffer and to provide an indication of how many bytes have been processed; and

a common program counter for the stack-based instructions and the register-based instructions, wherein the common program counter is incremented by the indication of the number of bytes processed.

148. (New) The CPU of claim 147, further comprising a common instruction cache for the stack-based instructions and the register-based instructions.

149. (New) The CPU of claim 147, wherein the instruction decode unit produces an indication for a variable stored in the register file

150. (New) The CPU of claim 147, further comprising a microcode unit coupled to the instruction decode unit to receive output therefrom.

151. (New) The CPU of claim 147, wherein the hardware accelerator produces an exception in respect of selected stack-based instructions.

152. (New) The CPU of claim 151, wherein the exception is processed using register-based instructions.

153. (New) The CPU of claim 147, wherein the instruction decode unit decodes multiple instructions received from the bytecode buffer in parallel.

154. (New) The CPU of claim 150, wherein the instruction decode unit generates a start address for the microcode unit.

155. (New) The CPU of claim 154, further comprising a mechanism to select the start address or an incremented start address for the microcode unit.

156. (New) The CPU of claim 153, wherein the instruction decode unit comprises multiple decoders.

157. (New) The CPU of claim 154, wherein the microcode unit produces a stack update indication to cause the instruction decode unit to generate a new start address for the microcode unit.

158. (New) A method for a central processing unit (CPU), comprising:

for register-based instructions, processing the register-based instructions in execute logic capable of processing the register-based instructions; and

for stack-based instructions, processing the stack-based instructions in a hardware accelerator into input the execute logic is capable of processing, wherein the hardware accelerator marks variables associated with the stack-based instructions in a register file as modified when the variables are updated as a result of the processing of the stack-based instructions to enable selective writing of the variables marked as modified to a memory.

159. (New) The method of claim 158, wherein processing the stack-based instructions comprises decoding multiple stack-based instructions in an instruction decode unit in parallel.

160. (New) The method of claim 158, wherein processing the stack-based instructions comprises generating exceptions in respect of selected stack-based instructions.

161. (New) The method of claim 159, wherein processing the stack-based instructions comprises generating a start address for a microcode unit in the instruction decode unit.

162. (New) The method of claim 161, processing the stack-based instructions comprises selecting the start address or an incremented start address for the microcode unit.

163. (New) The method of claim 158, further comprising storing the stack-based instructions and the register-based instructions in a common instruction cache.

164. (New) The method of claim 158, wherein processing the stack-based instructions comprises producing an indication for a variable stored in the register file.

165. (New) A method for a central processing unit (CPU), comprising:

for register-based instructions, processing the register-based instructions in execute logic capable of processing the register-based instructions; and

for stack-based instructions, processing the stack-based instructions in a hardware accelerator into input the execute logic is capable of processing, wherein the hardware accelerator generates a new virtual machine program counter (PC) due to a "JSR" or "JSR_W" bytecode by sign extending an immediate branch offset following the "JSR" or "JSR_W" bytecode and adding it to a virtual machine program counter (PC) of a current bytecode instruction, computes a return virtual machine program counter (PC) and pushes the return virtual machine program counter onto an operand stack.

166. (New) The method of claim 165, wherein processing the stack-based instructions comprises decoding multiple stack-based instructions in an instruction decode unit in parallel.

167. (New) The method of claim 165, wherein processing the stack-based instructions comprises generating exceptions in respect of selected stack-based instructions.

168. (New) The method of claim 166, wherein processing the stack-based instructions comprises generating a start address for a microcode unit in the instruction decode unit.

169. (New) The method of claim 168, processing the stack-based instructions comprises selecting the start address or an incremented start address for the microcode unit.

170. (New) The method of claim 165, further comprising storing the stack-based instructions and the register-based instructions in a common instruction cache.

171. (New) The method of claim 165, wherein processing the stack-based instructions comprises producing an indication for a variable stored in the register file.

172. (New) A method for a central processing unit (CPU), comprising:

for register-based instructions, processing the register-based instructions in execute logic capable of processing the register-based instructions; and

for stack-based instructions, processing the stack-based instructions in a hardware accelerator into input the execute logic is capable of processing, wherein the hardware accelerator performs sign extension for virtual machine SiPush and BiPush bytecodes and appends the sign extended data to an immediate field of a register-based instruction being composed based on the stack-based instructions.

173. (New) The method of claim 172, wherein processing the stack-based instructions comprises decoding multiple stack-based instructions in an instruction decode unit in parallel.

174. (New) The method of claim 172, wherein processing the stack-based instructions comprises generating exceptions in respect of selected stack-based instructions.

175. (New) The method of claim 173, wherein processing the stack-based instructions comprises generating a start address for a microcode unit in the instruction decode unit.

176. (New) The method of claim 175, processing the stack-based instructions comprises selecting the start address or an incremented start address for the microcode unit.

177. (New) The method of claim 172, further comprising storing the stack-based instructions and the register-based instructions in a common instruction cache.

178. (New) The method of claim 172, wherein processing the stack-based instructions comprises producing an indication for a variable stored in the register file.

179. (New) A method for a central processing unit (CPU), comprising:

for register-based instructions, processing the register-based instructions in execute logic capable of processing the register-based instructions; and

for stack-based instructions, processing the stack-based instructions in a hardware accelerator into input the execute logic is capable of processing, wherein the hardware accelerator performs sign extension for virtual machine SiPush and BiPush bytecodes and makes the sign extended data available to be read by the execute logic

180. (New) The method of claim 179, wherein processing the stack-based instructions comprises decoding multiple stack-based instructions in an instruction decode unit in parallel.

181. (New) The method of claim 179, wherein processing the stack-based instructions comprises generating exceptions in respect of selected stack-based instructions.

182. (New) The method of claim 180, wherein processing the stack-based instructions comprises generating a start address for a microcode unit in the instruction decode unit.

183. (New) The method of claim 182, processing the stack-based instructions comprises selecting the start address or an incremented start address for the microcode unit.

184. (New) The method of claim 179, further comprising storing the stack-based instructions and the register-based instructions in a common instruction cache.

185. (New) The method of claim 179, wherein processing the stack-based instructions comprises producing an indication for a variable stored in the register file.

186. (New) The system of claim 43, wherein the register file is used to store data for the register-based instructions, and an operand stack for the stack based instructions as a ring buffer.

187. (New) The CPU of claims 52, 54, and 56, wherein the register file is used to store data for the register-based instructions, and an operand stack for the stack based instructions as a ring buffer.